Extending MACS: Learning and Natural Language Processing for Enhanced Support of Contracting Officers in US Government Defense Research Contracting

March 2002

Bonnie Rubenstein-Montano[a†], Victoria Yoon[b], Stuart Lowry[c], & Teresa Wilson[b]

[a] The McDonough School of Business, Georgetown University
[b] Department of Information Systems, University of Maryland, Baltimore County
[c] Science Applications International Corporation

ABSTRACT. Contract procurement in US defense research contracting is a fairly personnel-intensive process. Recent advances in artificial intelligence, intelligent agents in particular, lend themselves to enhancing the status quo of US defense research contracting by automating, at least partially, aspects of the contracting process. A Web based multi-agent system, called the Multi-Agent Contracting System (MACS), has been developed as a first step in automation of the contracting process (Liebowitz et al., 2000; Rubenstein-Montano et al., 2001; Yoon et al., 2001). The research presented in this paper extends the capabilities of the original MACS system. Specifically, the existent Bayesian learning algorithm has been refined, reinforcement learning has been designed into the system to allow direct feedback from system users, and the grammar of the natural language interface has been expanded to better accommodate user needs.

1 INTRODUCTION

In U.S. defense research contracting, scientists expend significant effort to complete administrative details for contract acquisition, and often rely on the Defense Acquisition Deskbook for assistance. However, the current system requires human experts to respond to queries by scientists. Work on automating the process by developing a multiagent system to respond to scientist queries has been completed by several researchers (e.g. Liebowitz et al. 2000, Rubenstein-Montano et al. 2001, Yoon et al. 2001). The system, termed **M**ulti-**A**gent **C**ontracting **S**ystem (MACS), can be accessed at http://kmlab-01.ifsm.umbc.edu/macs/. MACS assists scientists with the pre-award phase of contracting. The work presented in this paper

† This work has been supported by the FY 2001 External Acquisition Research Program (EARP). The research was partially completed while Dr. Rubenstein-Montano was at the University of Maryland, Baltimore County.

extends the capabilities of the existing multiagent system by designing it to learn from system users and to interact with users via a natural language interface.

1.1 Learning

Traditional machine learning has developed a wide variety of algorithms for providing single-agent systems with learning capacity (Mitchell 1997). Among the main classes of algorithms for traditional machine learning are induction of trees and rules, learning in neural nets, system classifiers and genetic algorithms, reinforcement learning, Bayesian learning, case-based learning, logic-based learning, and some others. However, these algorithms do not apply directly when used in multiagent systems (MAS).

Learning in MAS has been studied by such researchers as Ayala and Yano (1998), who use agents in the area of computer supported collaborative learning, Vaario and Ueda (1996) look at modular learning in multi-agent environments, Norrie and Gaines (1995) who conceptualize learning on the Web through agents, and Rubenstein-Montano et al. (2001) who use agents in the area of defense contracting. There are also several researchers who have looked at specific learning techniques in multiagent systems such as Bayesian learning (Goldman & Rosenschein 1996; Sen & Sekaran 1996; Zeng & Sycara 1998) and reinforcement learning (Goldman & Rosenschein 1997; Prasad & Lesser 1999).

1.2 Natural Language Processing

Much of the theoretical and applied research in Natural Language Processing (NLP) comes from the field of Computer Science and Artificial Intelligence, although NLP also extends into linguistics, cognitive science, psychology and philosophy. In its most elemental sense, NLP refers to any technique used to process natural language.

NLP history spans five decades, beginning with the research of noted MIT linguist Noam Chomsky in the 1950s and 1960s to current day work in the 21st century. The traditional applications of NLP are text and speech processing, but NLP is evident in expert systems, intelligent agents, and smart interfaces. NLP has been applied to many application areas to include information extraction, information retrieval, machine translation, text summarization, speech synthesis, natural language generation and question and answering (QA).

The information explosion of the last 5 years has brought NLP back into the mainstream of research. Search and retrieval, particularly Web applications on the Internet, seem to generate the most interest and thus receive the lion's share of NLP research. In the past few large-scale Internet search engines have supported Natural Language querying, but most have returned to the traditional indexing techniques. Examples of current techniques include automatic words stemming, automatic identification of proper nouns and other classes of words, automatic phrase identification and automatic concept identification [Liddy, 1998].

Research in NLP continues, to include work on ontologies, semantic nets, and concept mapping, in additional to the traditional areas of algorithms, data structures, database, and statistical/stochastic methods. Example of recent studies include incorporating relevance feedback to improve performance (Iwayama, 2000; Vakkari, 2000), performance evaluation methodologies (Hersh et al, 2000; Buckley and Voorhees, 2000), event tracking for improved text categorization (Fukumoto and Suzuki, 2000; Yang et al, 2000), machine learning and probabilistic techniques (Kim, Hahn, and Zhang, 2000; Silva et al, 2000; Manivetz and Yousef, 2000; Petasis et al, 2000); distributed searching (Power et al, 2000), linguistic features and document clustering (Hatzivassiloglou, Gravano, and Maganti, 2000); cross-language retrieval (Sperer and Oard, 2000); question and answering (Prager et al, 2000; Voorhees and Tice, 2000); and Web content hierarchy and caching techniques (Lu and McKinley, 2000; Dumais and Chen, 2000).

2 THE MACS SYSTEM

The subsections below describe the MACS system prior to EARP FY 2001.

2.1 Description

The MACS system is a MAS developed for procurement and acquisition of defense contracts. Specifically, it is designed to assist contracting officers with the pre-award phase of contracting and procurement. The system architecture consists of nine agents — a User agent, a Facilitator agent, a Natural Language agent, a Machine Learning agent, and five specialty agents. The specialty agents are encoded with domain knowledge about the five general areas of expertise required of contracting officers, and the user agent interfaces with contracting officers. Interaction between contracting officers and the system occurs through either keyword searches or natural language queries. As shown in Figure 1, the MACS architecture implements a typical three-tiered brokered architecture. The Facilitator agent coordinates agent activities and communicates with the agent(s) capable of responding to an incoming query.

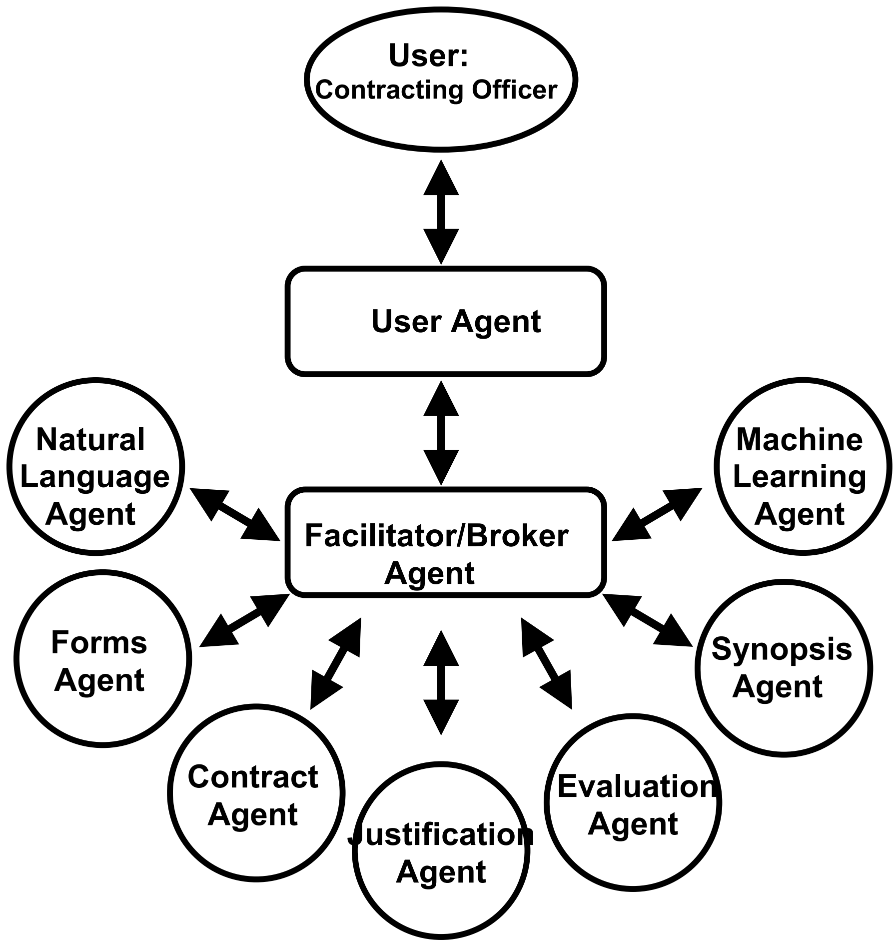


Figure 1: Agent architecture and communication channels

The user agent interacts with the user/contracting officer to welcome the user, ask what pre-award questions the user has, and serve as the interface between the user/contracting officer and the other agents in the system (Liebowitz et al. 2000). Two business logic threads have been designed into the User agent. One thread supports the Natural Language capability of the system, and the other supports the keyword search capability of the system. The User agent sends incoming user queries to the Facilitator agent, which is responsible for communicating with all of the other agents in the MACS system as illustrated in Figure 1.

Queries submitted by users are forwarded, by the Facilitator agent, to the Machine Learning (ML) agent, the ML agent implements Bayesian learning and creates an action plan, and that plan is then issued back to the Facilitator agent for completion. In MACS, the action plan is essentially a determination of which specialty agent(s) should be contacted to respond to an incoming query. The facilitator completes the action plan by performing the necessary low-level communication between the specialty agents. These communications lead to solutions being sent from a specialty agent (or agents) to the Facilitator agent and then from there to the User agent. As solutions to a query are collected, the ML agent updates its internal tables, making note of which agents responded to which user queries. This information is used to calculate the response plan for similar queries in the future.

The five specialty agents in the system relate to the pre-award phase of a contract and include the *Forms, Justification, Evaluation, Synopsis*, and *Type of Contract* agents. The Forms agent identifies the forms needed to complete a procurement request package. The Justification agent indicates situations where a justification and approval is required to complete a procurement request. The Evaluation agent provides guidelines for evaluating proposals. The Synopsis agent identifies the type of synopsis for a given procurement request. Lastly, the Type of Contracts agent identifies the type and nature of a contract based on conditions such as the source of contract, the nature of the work, etc.

Each agent in MACS contains a rule base and has explicit goals. Its rule-base describes how to achieve the goals under varying circumstances. The specialty agents respond to incoming queries by presenting necessary information and/or requirements for contracting officers. For example, the Evaluation Agent can assist a contracting officer with information regarding how to evaluate a project and what criteria or weights to use for evaluation of a contract. If a contracting officer has a question regarding "determining weights on evaluation criteria," the Evaluation agent will reply with "You can develop your own weights on technical, qualifications, and cost criteria. Generally speaking, a weight of 40 percent (out of 100%) is given to cost." (Liebowitz et al. 2000).

The knowledge contained within each specialty agent is independent of the knowledge contained within the other specialty agents. Thus, coordination between the specialty agents is not required for the current implementation. However, each specialty agent does coordinate with the user agent in order to answer queries. In the original system (Liebowitz et al. 2000), the user agent broadcast messages to all specialty agents. The learning capability that is now part of MACS allows the user agent to learn which specialty agent(s) should receive incoming messages. The user agent asks the ML agent to determine which specialty agent(s) should receive the query. The ML agent makes this determination probabilistically, by means of Bayesian learning.

2.2 Learning in MACS

The existing MACS system, which resulted from EARP 2000, uses Bayesian learning applied in the ML agent. Here, learning involves learning which specialty agents should receive incoming messages. The approach is similar to that of Heckerman and Horvitz (1998) where user goals are inferred from user queries using a naive Bayesian classifier. However, the

Heckerman and Horvitz (1998) approach allows for free-text queries whereas MACS is designed for both free-text, Natural language queries and keyword searches.

Essentially, the user agent employs a Bayesian model to identify which specialty agent(s), a_i, are most likely to respond correctly to a query given the evidence, e, appearing in the query, q. The Bayesian learning procedure is as follows:

1. User enters a query, $q_{incoming}$
2. Use Bayesian reasoning to determine which a_i should receive the query.
   For each a_i (i = *evaluation*, *synopsis*, *justification*, *forms*, *type of contracts*):
   - ◆CALCULATE the percentage of time each keyword appears in all $q_{existing}$ (e.g., evaluation criteria appears in 80% of existing queries sent to the $a_{evaluation}$)
   - ◆CALCULATE probability(evidence/a_i) by multiplying all percentages calculated in the immediately preceding step that correspond to $q_{incoming}$. Probability(evidence/a_i) represents the likelihood that a query actually corresponds to the domain knowledge of that a_i. This is a causal relationship from the cause (a_i) to the effect ($q_{incoming}$).
   - ◆USE the Bayesian formula by (a) multiplying the prior probability, P(a_i), by probability(evidence/a_i), (b) sum all calculations from (a), and (c) divide each individual result from (a) by (b). This will give the probability(a_i/evidence). P(a_i) represents the likelihood $q_{incoming}$ should be sent to a particular a_i given no evidence. At time 0, P(a_i) = 0.2 for all i. Probability(a_i/evidence) is the posterior probability distribution of a_i given the evidence. This probability assesses the likelihood that a specialty agent will answer $q_{incoming}$ based on the evidence provided by $q_{incoming}$ itself.
   - ◆DIVIDE probability(a_i/evidence) by P(a_i)

   After completing the calculations for each a_i
   - ◆IDENTIFY the result with the greatest value
   - ◆SEND $q_{incoming}$ to the corresponding a_i
   - ◆ADD $q_{incoming}$ to the list of $q_{existing}$ for the corresponding a_i

2.3 Natural Language Processing in MACS

The NLP agent in MACS is the Definite Clause Grammar-Natural Language (DCG-NL) in OAA, which is designed to parse incoming natural language queries. Through the use of a logic-based grammar called Definite Clause Grammar (Pereira & Warren, 1980), the DCG-NL translates incoming English queries into expressions using an agent communication language used in OAA, called Interagent Communication Language (ICL).[1]

ICL expressions are internal OAA representations of the natural language query that the agents can act upon. Each parsed word or phrase must be registered in the vocabulary of the appropriate agent. The name of each agent corresponds to the functor, or a term that represents a particular function. For MACS, the functors are *"Forms", "Justification", "Types of Contract", "Synopsis", and "Evaluation"*. In cases where there may be multiple expressions synonymous to the functor, an alias was created so that expressions could be resolved to one functor. For example, for the Evaluation agent, "evaluation" is the function, so the functor is the word "evaluation". Phrases such as "evaluation weights" and "evaluation criteria" were aliased to "evaluation" so that any ICL expression with the functor "evaluation" will be sent to the Evaluation agent. There also are nouns, adjectives, and verbs that are used by all agents. These words, along with their lexical categories, are combined in one list called "Common". Each word is XML-tagged with its appropriate part of speech. This allows the DCG-NL, powered by the linguistic content of its grammar, to recognize the lexical information so that it can effectively parse the query. Examples of parsed outputs are listed in Table 1.

[1] ICL is a logic-based declarative language that expresses high-level, complex tasks and natural language expressions. SRI chose a logic-based language because it was important that the language the agents spoke could represent and easily translate back and forth to human language (The Open Agent Architecture™, 2000).

Table 1: Examples of Parsed Outputs

```
forms('capital equipment',['major procurement'],)

wh(var(forms([]),go([in(some('PR package'([for(some(procurement(['capital equipment’,
    major]))))]))),subject(var()]))

contract(arrangement('labor hour'),[])

evaluation(proposal([unsolicited,for(award('sole source'))]),[])

synopsis('I',procurement(['ADP',under(count(50000,dollars))]),[on(schedule('GSA'))])
```

When a query is parsed, the Facilitator agent sends the parsed output to a specialty agent capable of handling the query. That agent will "fire a rule" from the knowledge base and return a relevant piece(s) of information to the user; that is, when the condition stipulated in the rule is met, the user is provided with the answer to his/her natural language query. An answer is returned if any one or more rules fire within the specialty agent.

3 COMPLETED WORK for EARP 2001

3.1 Learning in MACS

For EARP 2001, two key extensions to the learning capabilities have been designed into the MACS system. The first enhancement is directly in-line with the FY 2000 EARP work. In the 2000 system, Bayesian learning was built into the system via the machine learning (ML) agent. Here, the system learned which specialty agent should receive an incoming query. However, the original implementation allowed only one specialty agent to receive each incoming query. There are cases where several specialty agents may have partial answers to a query.

Thus, the system has now been upgraded to allow the ML agent to learn which agent(s) to whom incoming queries should be sent. It is possible that only one agent will receive a query, but it is also possible that more than one will receive a query if the ML agent deems it possible that more than one agent has knowledge relevant to the query. The enhancement works as follows:

1. Use the Bayesian probabilities as they are currently calculated to sort the specialty agents in descending order of probabilities.
2. Send the query to any and all agents that have a Bayesian probability equal to that of the first ranked specialty agent.
3. If the probabilities are not all equal, check to see if the second ranked agent has a probability within 1/100th of 1% of that of the first ranked agent. If so, send the query to both the first and second ranked agents.
4. Otherwise, if the probabilities are not all equal, and the second ranked agent does not have a probability within 1/100th of 1% of that of the first ranked agent, send the query to only the first ranked agent.[2]

The second extension to learning in the MACS system involves feedback, or reinforcement, learning in the User agent. Reinforcement learning (Jerbic et al. 1999; Kokar & Reveliotis 1993; Maclin & Shavlik 1996; Prasad & Lesser 1999; Shen 1994) involves agents acquiring new knowledge (i.e., learn) by acquiring feedback from previous experiences and the environment. A reinforcement signal results from an agent's actions. However, specific actions are not provided; and the agent learns to improve performance based on the signal(s) (Jouffe 1998).

In the MACS system, reinforcement learning occurs in the Natural Language (NL) agent, and the natural language interface and learning capabilities are integrated to some extent. Essentially, MACS learns what a user is trying to ask based on similar past questions. When a

[2] 1/100th of 1% was selected for initialization purposes. Once the system is further tested, this value may change, depending on probability values (i.e., if probabilities rarely get that close, or are always that close, the value will be adjusted accordingly).

user inputs a new natural language query, it either does or does not parse. If it parses, MACS functions as it did at the end of FY2000.

The Gemini parser added to the system in FY2001 is equipped to handle a finite number of grammatical constructs, but users are able to ask questions many different ways. Therefore, we have built a web application that allows users to ask a question several different ways for when questions cannot be parsed and until a satisfactory answer is located.

If the query does not parse, reinforcement learning is invoked because the query cannot be resolved until it is parsed by the NL agent. There are two steps taken by MACS if a query does not parse. First, a cache is searched to see if the same query was asked previously. If so, the results presented previously are presented again. In this case, the cache operates as a hash table. As sentences that cannot be parsed are submitted, they will be associated with sentences that have been parsed. If the user submission is matched to a previously unparsed sentence in the cache, the submission that cannot be parsed will be exchanged with the one that can be parsed. The sentence that can be parsed will then be submitted in-lieu of the original sentence. This technique reduces the burden on the NL agent for learning new vocabulary and grammar by allowing past users to create “synonymous sentences”.

If the same unparsable query is not present in the cache, a second step is taken. Here, the user is asked, by the User agent, to "categorize" the question, and try to re-phrase it. Users can categorize the query according to the five specialty agents. Once rephrased, the original query (hidden field on form) and the new one are sent to the User agent. From here, two things may happen:

1. If the rephrased question parses, the rephrased question then serves as the feedback for the original question. If, in the future, another user asks the same original, unparsable query, MACS will have learned from the user who first asked the question what is really being asked. Instead of requiring this new user to rephrase the query, MACS remembers the version of the question that can be parsed by the NL agent, and submits that query on the behalf of the user.

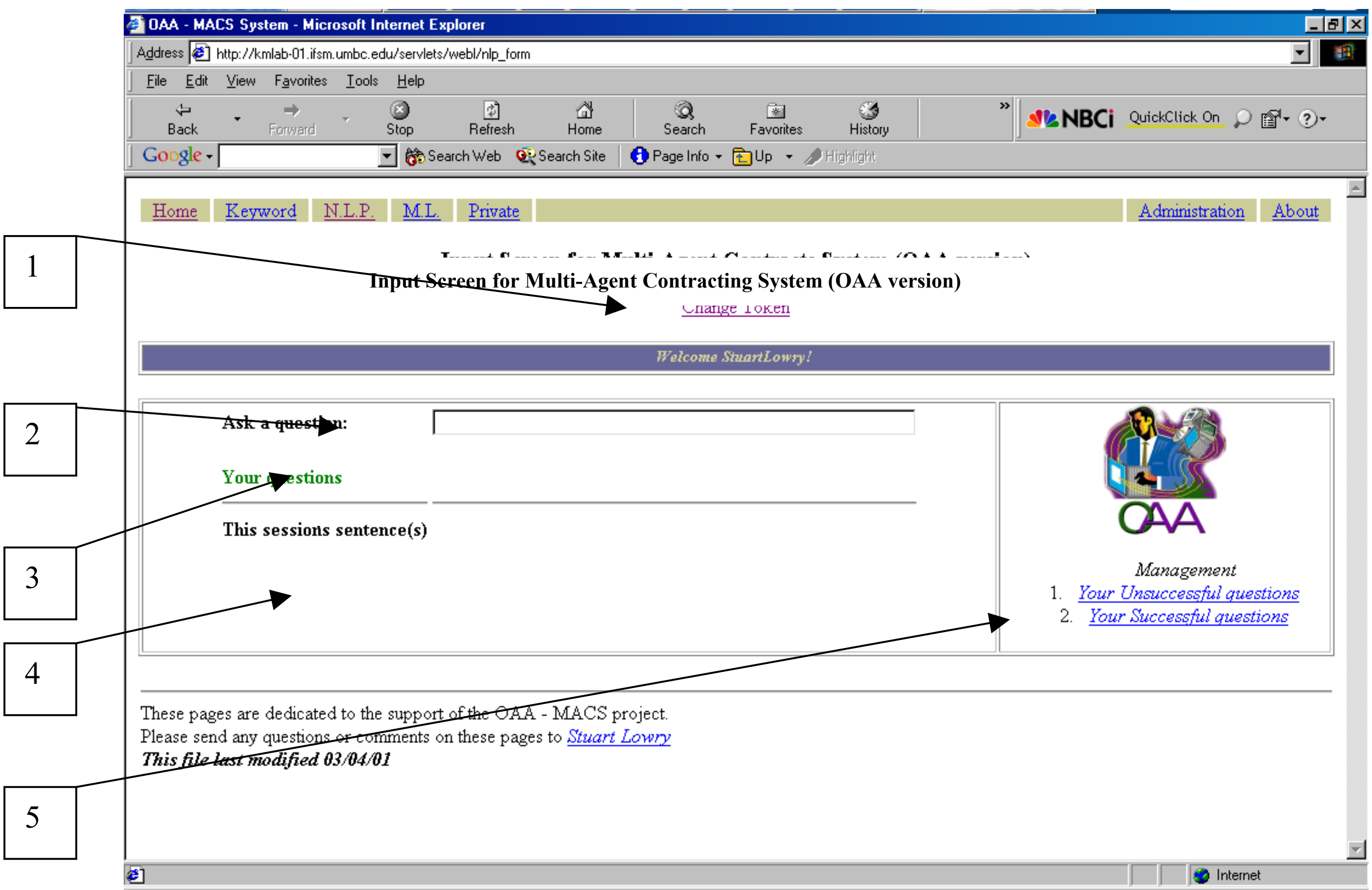
2. If the re-phrased query still does not parse, all previously asked and answered queries in the same "category" are presented to the user in the form of a pick list. The user can then select one of the resolved queries. If a previously resolved query is selected, the series of unanswered, rephrased queries are linked to the resolved query. This serves as feedback so that if the same, unparsable query is submitted in the future, MACS will resolve it without requiring the user to complete any of the aforementioned steps. This serves as a sort of "synonomous sentence cache (SSC)."

The feedback cycle continues until the query is resolved. Once an answer is identified, all sentences supplied during a given session will be presented to the user with the retrieved answers. The user will be able to select the earlier session sentences as being synonymous. These synonymous sentences will be used as the basis of reinforcement learning whereby MACS will learn how to provide responses to queries that cannot be handled by the Gemini parser. When a subsequent user session occurs, the synonymous sentences will be tied to the parsable question from some prior session. This technique increases the breadth of sentences supported by the MACS system, without having to modifying the grammar or the vocabulary for the Gemini NL parser.

3.2 Reinforcement Learning Example

In this section we adopt the approach taken by Maulsby & Witten (1997) to review learning in the MACS system. An example is reviewed instead of presenting a statistical analysis of results. A series of screen shots below illustrate the process. The user is presented with a web page that will manage a "session". The user controls a session in order to drill through the knowledgebase to discover an answer to their question.

Screen 1: User "StuartLowry" arrives at the NLP Submission form



1. Change Token – This hyperlink allows users to set a cookie so that their interactions can persist. All submissions into the system are tied to this cookie. It is not a security feature but rather a means to identify unique users. Any identifier can be used.

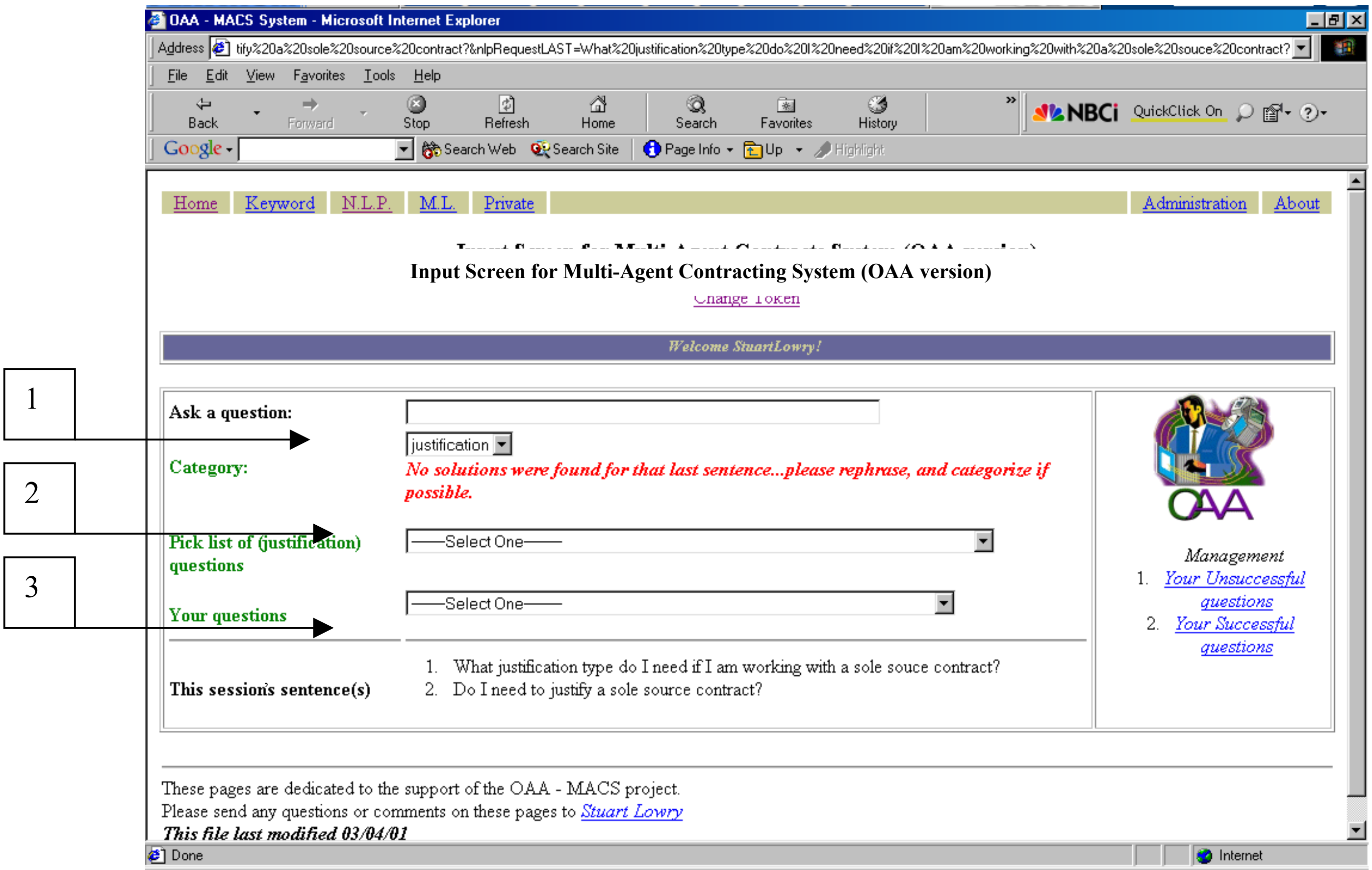
2. Ask a question – The user freely types any question he wishes to submit to the system.

3. Your Questions – All questions submitted by the users are available. The idea is that users frequently ask the same questions. As the rule base behind MACS evolves (new contract restrictions are created, new laws are enacted, etc), a user may wish to come back to MACS and see if the answer to a question that was asked before, has changed. They can quickly select one of the many questions they have already entered from a past session.

4. This Session's Sentence(s) – This area is used to display the unanswered questions of the current session. It acts as a sort of "shopping cart" of sentences, where the system remembers all sentences input during the current session. Then the user can match [select] unanswered questions with the one that asks the same basic question but that returned a meaningful response.

5. Management – The hyperlinks in this section will allow users to remove questions from the persistent cache that appear in item #3.

Screen 2: User "StuartLowry" supplies an un-answerable question[3]



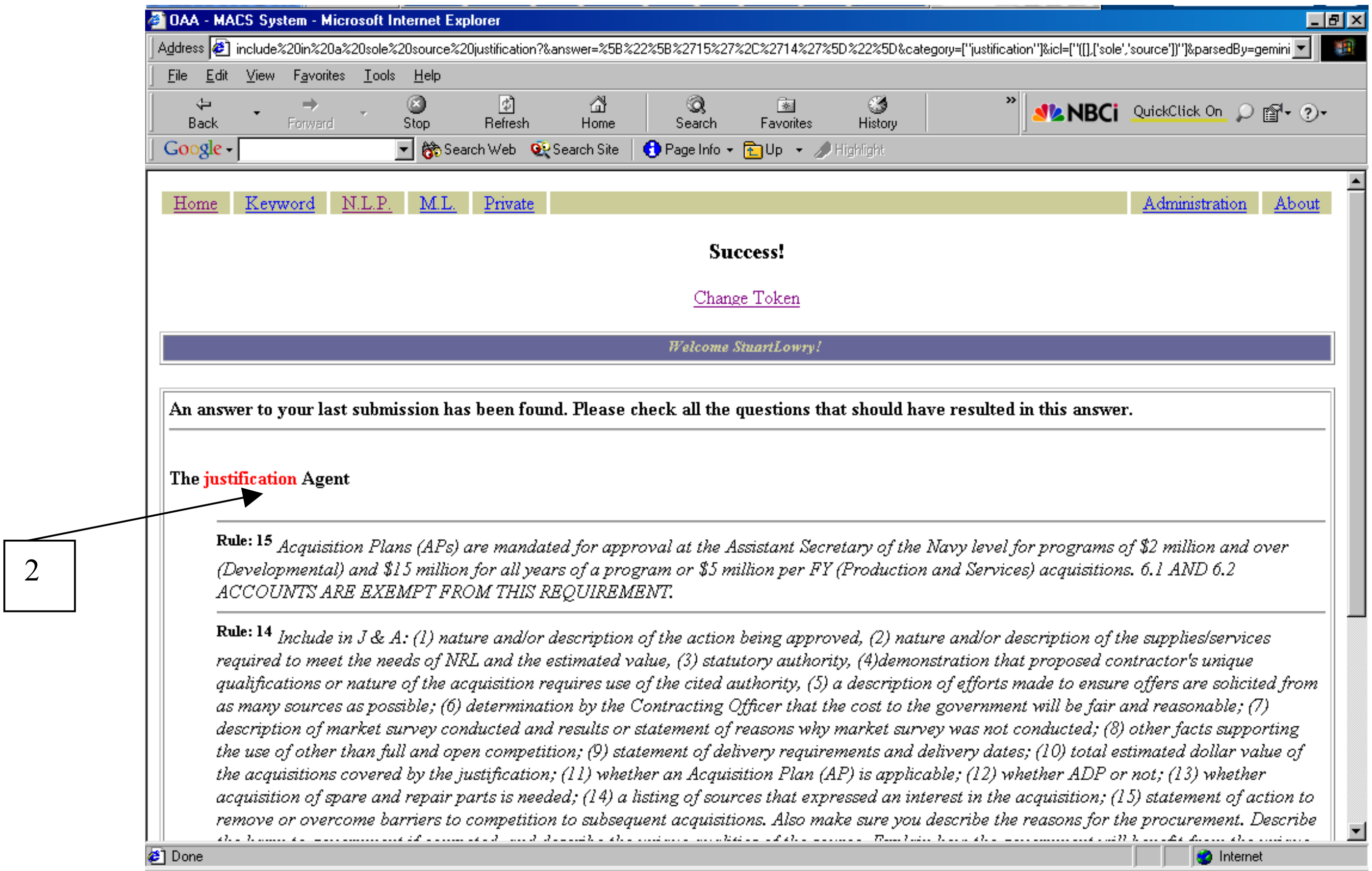
1. Category – When MACS is unable to answer a question, the user can categorize the problem. The categories are directly related to the specialty agents that are running in the system. In this case, the user has selected the justification category.

2. Pick list of questions (justification) – This pick list contains all of the questions that have been answered to date by the justification agent. The user can scan this list to see if their question has already been asked and answered. If they select a question here it will be as if they typed it into the form themselves. When this question is answered again the user can make the original question a synonymous question and thus increase the scope of the system. This is the reinforcement learning that occurs in the system as a result of user input.

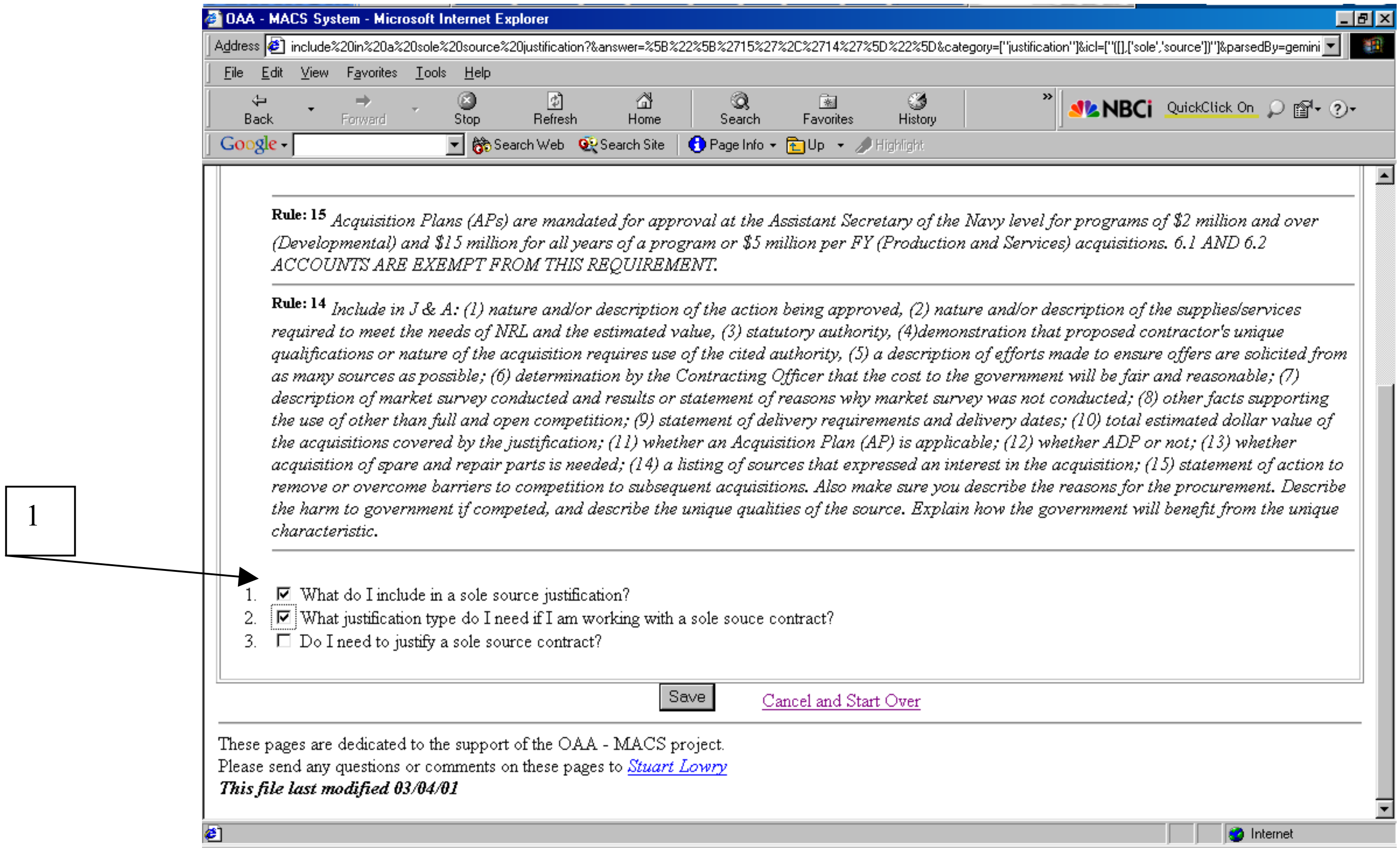
3. The session space is updated to reflect this session's submissions.

[3] Actually, this question can be answered, but the Gemini parser was turned off for illustration purposes.

Screen 3: User "StuartLowry" submits a sentence that is answerable: *What do I include in a sole source justification?*



Screen 3 (con't)



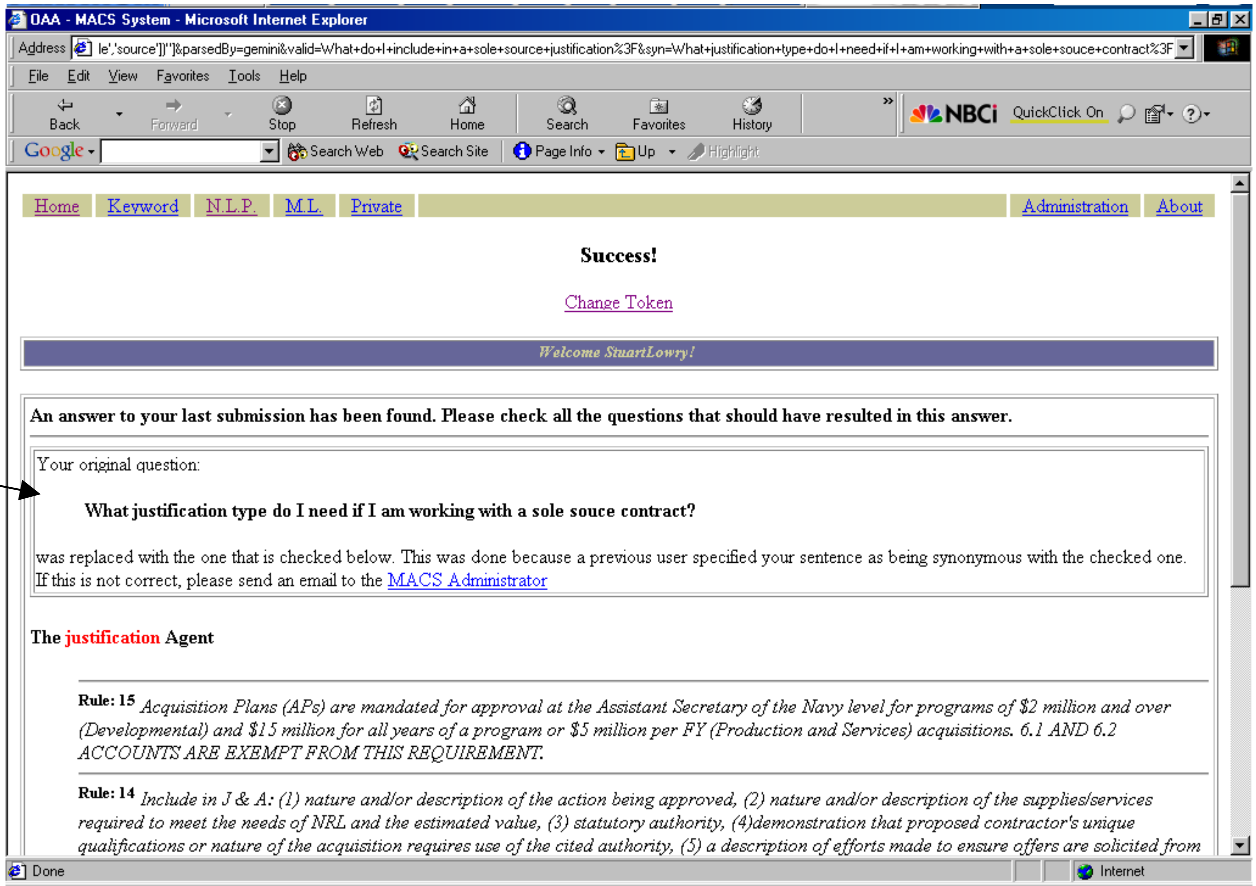
1. #1 is checked because it is the sentence that was successfully answered. The user can now check all other sentences that are "synonymous". This feedback serves as the input, or reinforcement, needed for learning in the user interface part of the system so that MACS can learn how to provide responses for sentences that cannot be handled by the Gemini parser.

2. #2 & #3 are the previous attempts to solve the user problem.

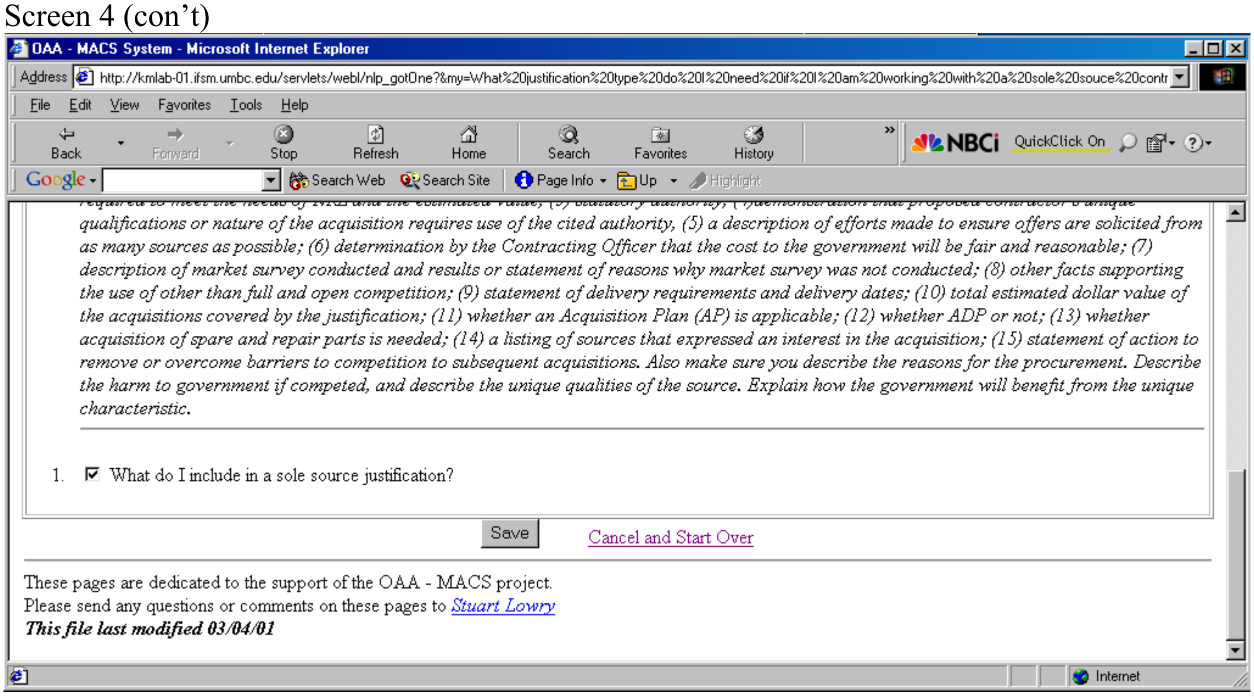
3. If the save button is selected, each of the checked items will be saved (#1 & #2). The additionally checked sentences (#2) will be synonymous to the sentence that was successfully answered (#1). Therefore, when a future user submits the #2 sentence, this same answer will be returned rather than the unsuccessful screen.

Screen 4: A new user session occurs and the previously unanswerable question that is now a synonymous sentence is submitted.

1



Screen 4 (con't)

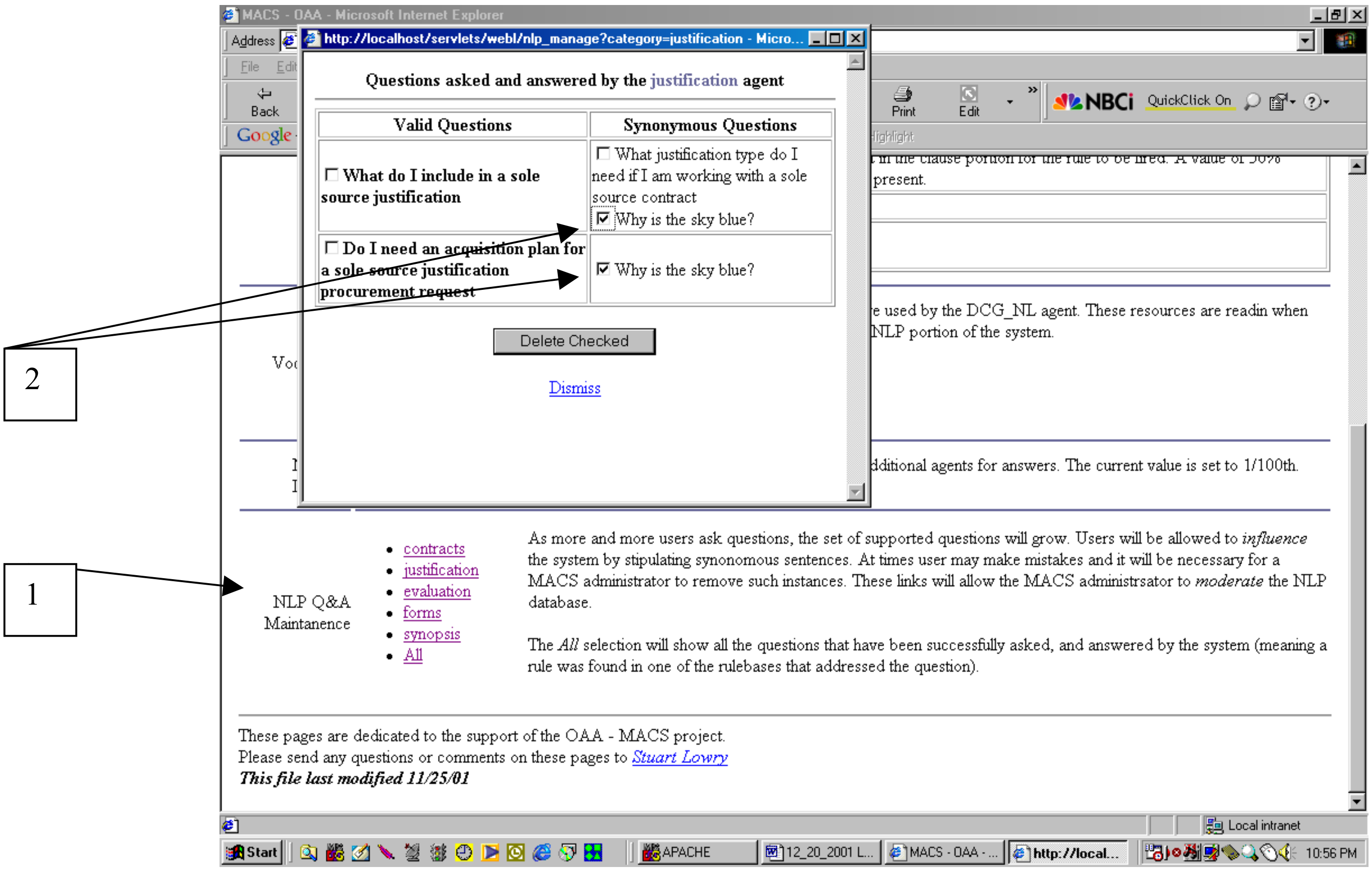


1. The user is notified that while an answer was found, this sentence was replaced with its supported counterpart. It should be noted that the counterpart is resubmitted to the system. The supported sentence is resubmitted in case the rulebase has been modified. This is important because MACS does not manage a cache that ties answers to questions. Instead, questions are evaluated each time they are input to save memory.[4] Once a question is input that does not parse but that has been deemed equivalent to a sentence that does parse (by a user), the sentence that does parse will be resubmitted on the user's behalf. This maintains system flexibility in that answers can evolve over time without interrupting the system since answers are not hard coded into question-answer combinations.

Screen 5: Administration

In order to oversee the user-supplied information, a MACS administrator can selectively eliminate any question that has been marked as synonymous. Because users can broaden the scope of the MACS system, there is always the potential for users to enter inaccurate information. Thus, the MACS system administrator retains ultimate control over which sentences are deemed equivalent so that clearly inappropriate user selections such as those in the screen below can be removed.

[4] This concept is similar to that of the Internet, where the location of information is of utmost importance. Rather than trying to physically store all information on a single PC, server, etc., locations are retained for access when necessary which is more efficient from a storage perspective.



1. Each of the specialty agents has a synonymous sentence cache. Access to this page is restricted to MACS Administrators.

2. The checked items will be permanently removed from the system.

3.3 Natural Language Processing in MACS

In order to improve the performance of the natural language interface of the User agent, the DCG-NL agent has been replaced by another parser called ATTAIN. ATTAIN is a package of natural language OAA agents which provides parsing and translation of English sentences into ICL messages that other agents can use. The difference between ATTAIN and the DCG-NL is that ATTAIN uses a unification grammar while the DCG-NL used a Definite Clause Grammar.

Unification grammar means that grammatical categories incorporate features that can be assigned values; so that when grammatical category expressions are matched in the course of parsing or semantic interpretation, the information contained in the features is combined, and if the feature values are incompatible the match fails. This parser technology provides a more expressive ICL representation and other advantages as well. (The Open Agent Architecture, OAA, 2000.)

The ATTAIN suite comes with four OAA agents, two of which are used in MACS. The attain_nl_agent takes the incoming query and transforms it into a logical form (LF), which represents the meaning of the sentence. Then, the nl_icl_agent takes the LF and transforms it into an ICL representation. For example, for the query "What contract do I need for labor and material only?" the LF is:

(wh((), quant(wh,contract),vpred(need,subject(),object(()),
ppred(for,quant(some,[labor,and,material,only],[]))]))))

while the ICL is:

(wh(),contract,need(me),[for([labor,and,material,only])]))

In other words, the LF represents the grammatical construction of the query with linguistic markers such as wh, quant, vpred, subject, object, ppred. The ICL representation translates and simplifies the LF.

For EARP 2000, significant modifications were made to the original user queries because the DCG grammar was limited and could not successfully parse the queries. This shortcoming was the impetus for replacing the DCG-NL parser with ATTAIN. ATTAIN incorporates most of the sentence types that the DCG-NL can parse, but it has more sentence types that can be parsed. ATTAIN has a few features that DCG-NL does not, including:

- inflected forms of regular nouns and verbs (singular/plural, present/past tense) do not need to be individually entered as vocabulary items. ATTAIN analyzes morphology;
- the "double object" construction is allowed, whereby both direct and indirect objects that are noun phrases can be parsed, resulting in successful parses of longer, more complex sentences;
- the arguments of a verb are listed in order, so subsequent processes requiring linguistic analysis could be performed more easily.

Table 2 presents a sample set of the sentence types that ATTAIN can parse but the DCG-NL could not.

Table 2: Types of Sentences Parsed only by ATTAIN

Types of Sentences
Which contract type do I submit if my proposal deals with university research?
How do I determine the scoring of evaluation criteria for competitive solicitations?
What is the procedure for synopsizing when submitting an unsolicited proposal?
Do I need an acquisition plan if I am providing a sole source justification?

In general terms, ATTAIN allows for both active and passive voice constructions, extensive use of modals (should, could, would), and longer verb predicates (longer lists of noun phrases and prepositional phrases after the verb). These seemingly simple features greatly enhanced the type of queries that are encountered in the contracting domain.

ATTAIN still has its limitations, however, since it is still biased toward the particular domain for which it was built. It does not handle conditional phrases to the extended MACS needs to, such as “If I am dealing with a major procurement contract that involved non-library materials and non-equipment purchases, what forms should I use?” It has problems handling numbers in that numbers must function as modifiers, such as 5 hours, and certain special characters cannot be used, such as ‘&’ or ‘$”. This was overcome by modifying the queries into parseable phrases, rendering multi-term tokens that include numbers into single-term tokens by using underscores “_” between the terms, such as DD_Form_1498, and expanding ‘&’ and ‘$’ to ‘and’ and ‘dollars’, respectively.

3.4 Information Retrieval

The information retrieval portion of MACS has been upgraded also. Information retrieval (IR) within MACS occurs after the parsing of the natural language user submission. IR binds the ICL resulting from the natural language agent, to the actual rules in the specialty agents. The IR process is comprised of steps that include reducing the ICL into a series of words, elimination of stop words, and Boolean matching the terms in the conditional portion of the specialty agent rules. First, the ICL resulting from the NL agent is stripped down.

1. All of the special characters are stripped out of the ICL (parentheses, commas, double & single quotations, brakets).

2. All of the prolog style lists are expanded to a series of terms. (e.g. [‘up the hill’,’Ed’,’Cameron’] gets turned into “up the hill Ed Cameron”.

The second step is to remove the stop words. The stop words are managed by the MACS System Administrators. A web form was developed to allow system administrators to add, and remove words from the stop word list. The stop words are those deemed by the Administrators of the MACS system as beiong statistically insignificant. Therefore, whenever these words are present in the resulting ICL, they are removed from further consideration. The final step performs the Boolean matching function to the individuals rules in the specialty agent. This piece of the IR occurs in the specialty agent itself. This logic occurs here because it is considerd to be the “value added” by the specialty agent. Each agent is capable of applying a unique criteria to match the word list to its individual rules. At this time the distinction between the specialty agents is the rulebase knowledge and not their individual functionalities.

Therefore, each specialty agent applies the same matching algorithm. The matching algorithm consists of a boolean matching percentage, and a whole word match requirement. Instead of an implicit Boolean “AND”, a percentage scale was used. 1% was equivalent to the implicit “OR” and 100% was equivalent to the implicit “AND”. Although performance using the implicit “AND” was good, it was too inflexible because the IR was performed on only perfectly exact matches. But, an implicit “OR” was too unconstrained, resulting in too many irrelevant rules to be fired. There really was no good way to control the search in 2000. This was the motivation behind the percentage scale. The whole word matching requirement assisted in narrowing the result set to the proper rule. As each word in the word list was matched in the clause portion of a particular rule, it was flagged as matched. Each term in the word list was treated with the same weight. At the end of the matching processess, the number of hits was divided by the number of terms present in the word list and multiplied by 100. If this value is greater than or equal to the pre-set Boolean percentage, then the rule will be considered an answer to the question. An end-to-end example of the IR approach is detailed in table 3.

Table 3: Information Retrieval Example

1.	User Submission	"Which type of contract do I need for a labor hour arrangement?'
2.	NL output (ICL)	Solve(wh(var(_9064),[type,of,contract],need(me,var(_9064),[for([labor,hour,arrangement])])),[])
3.	Strpped ICL	wh var type of contract need me var for labor hour arrangement
4.	Word list supplied to specialty agents (stop words removed)	'contract','labor','hour','arrangement'
5.	Sample rule in the "contract" agent	<rule name="5"> <condition> <or> <clause>labor hour</clause> <clause>time and material</clause> <clause>materials</clause> <clause>indefinite delivery</clause> </or> <or> <clause>arrangement</clause> </or> </condition> <answer> The type of contract is cost-plus-fixed-fee </answer> </rule>

Table 3 (con't.)

6.	Matching Output	<condition> <or> <clause>**labor hour**</clause> <clause>time and material</clause> <clause>materials</clause> <clause>indefinite delivery</clause> </or> <or> <clause>**arrangement**</clause> </or> </condition>
7.	Statistical comparison	3 hits / 4 words * 100 = 75% the present Boolean Pecentage was set to 50% so thefore, this rule is valid.

3.5 Results

Thirteen (13) out of 23 queries from EARP 2000 and 13 new sentences were used to test the performance of NLP and IR. Only those queries that had applicable rules in the rule base were used. All 26 sentences (100%) were successfully parsed. Moreover, fewer queries needed to be modified because ATTAIN is more robust than the DCG-NL. The following are examples of the queries, their ICL expressions, and final parses for IR:

What do I evaluate for a sole source unsolicited proposal?
oaa_Solve(wh(var(),thing([]),evaluate(me,var(),[for([sole,source, unsolicited,proposal])])),[])
(['evaluate'],['sole','source','unsolicited','proposal'])

What evaluation do I need for a sole source unsolicited proposal?
oaa_Solve(wh(var(),evaluation,need(me,var(), [for([sole,source, unsolicited,proposal])])),[])
(['evaluation','sole','source','unsolicited','proposal'])

When do I synopsize an ADP procurement under 50000 dollars on the GSA schedule?
oaa_Solve(wh(var(),time([]),synopsize(me,procurement([adp]),[under(count(50000,dollar([on([gsa,schedule])]))),to(var())])),[])
(['synopsize','procurement','under','dollar'],['adp','50000','gsa','schedule'])

What type of synopsis do I need for an ADP procurement under 50000 dollars on the GSA schedule?
oaa_Solve(wh(var(),type([of(synopsis)]),need(me,var(),[for(procurement([adp,under(count(50000,dollar([on([gsa,schedule])])))]))])),[])
(['procurement','under','dollar'],['synopsis','adp','50000','gsa','schedule'])

Do I synopsize an ADP procurement under 50000 dollars on the GSA schedule?
oaa_Solve(synopsize(me,procurement([adp]),[under(count(50000,dollar([on([gsa, schedule])])))]
(['synopsize','procurement','under','dollar'],['adp','50000','gsa','schedule'])

How is scoring of evaluation criteria determined for competitive solicitations?
oaa_Solve(be(how,scoring([of(evaluation([rel(var(),determined(criteria, var(),[for([competitive,solicitations])]))]))]),[]),[])
(['scoring','determined'],['criteria','competitive','solicitations'])

What is the scoring of evaluation criteria for competitive solicitations?
oaa_Solve(wh(var(),thing([]),be([scoring,of,evaluation,criteria,for, competitive,solicitations],var(),[])),[])
(['scoring','evaluation','criteria','competitive','solicitations'])

The performance of NLP was tested on the basis of the percentages of parsed words presented in the condition part of a rule. It was determined to use the 50% as the threshold, whereby half of the parsed words needed to be present in the rule base to fire a rule. Ground truth rules, which are those that the domain expert had determined to be the “correct” rules, were identified and used to evaluate performance. Table 4 presents the summary of NLP/IR performance results.

Table 4: Summary of Results

Types of Result	**Hit Ratio**
Queries returning a rule	26 out of 26, 100%
Queries returning only the ground truth rules	8 out of 26, 31%
Queries returning more rules in addition to ground truth	18 out of 26, 69%

Although there was a significant increase in recall or coverage compared to MACS 2000 results, precision did not seem to have been affected since all of the ground truth rules were fired. There are three possible reasons for these performance results. One is due to the 50% search percentage. Since the search was not restricted to exact term matches, there were more rules in the rule base that would have the search terms in them. In other words, the search space just increased. The second is due to the fact that more than one agent responded with rules (for 12 of the queries). Strictly speaking, these additional rules could be considered "irrelevant" hits, but these additional rules provide useful information to the user, even if they were not answered by the "most correct" agent and were not the "most relevant" information. For example, the Contracts agent successfully answered a contracts-related query, but additional information provided by the Synopsis agent could be something the user would need to know later if the user subsequently needed to submit a synopsis. This could be a time saver in that the user would not have to query MACS a second time for a Synopsis-related question.

The third is due to the terms themselves. There was a significant number of additional rules returned for 2 queries: "*Do I need an acquisition plan for a sole source justification procurement request?*" (12 total rules from 3 agents), and *"What forms do I use for a major capital equipment procurement?"* (23 rules from 2 agents). In these cases, the parsed outputs are, "*'acquisition', 'plan', 'sole', 'source', 'justification', 'procurement'*", and "*'forms', 'major',* *'capital', 'equipment', 'procurement'*", respectively. For both queries, the term "procurement" is problematic. Although "procurement" is a ubiquitous term in the rule base, it has enough semantic content that it cannot be considered as a stop word. The term "major" also is another ubiquitous one, particularly in the Forms rule base. So, not only is the search technique and its effect on discriminating among agents an important feature, the semantic weight and frequency of occurrence of the actual search terms in the rule base, equally are critical features that affect MACS performance.

4 REMAINING WORK

None

5 CONCLUSIONS

The features of the MACS system presented in this report suggest ways in which multiagent systems can become increasingly useful for human users. This is particularly promising for acquisition research because of its current heavy reliance on people, which are an expensive and valuable resource. In previous EARP research, learning was built into MACS as a "proof of concept", but now significant degrees of learning, along with a natural language interface, have been designed into the system so that MACS can be a truly useful tool for human users. Despite the advances of the MACS system illustrated in this report, field testing and revision of the system is necessary before it can be used by contracting officers. Details of future work are outlined below.

5.1 Future Work

There are three primary areas for future work. The first two areas involve automatic updating of vocabulary/synonyms in the MACS system, and the last involves making the system useful in practice.

1. No rule is fired or no specialty agent is identified
This can be handled in one of two ways. (A) The User agent will parse the sentence and ask the user for synonyms for key words in the sentence. Synonyms will then be added to the grammar automatically. (B) The User agent will ask the user to rephrase their question.

2. Words are input to the NL interface that MACS has not seen previously
The User agent will parse the sentence and ask the user for synonyms for key words in the sentence. Synonyms will then be added to the grammar automatically.

3. A group of potential system users must be identified to use the system and provide feedback about problems and necessary changes.

REFERENCES

1. Ayala, G., and Yano, Y. (1998) A collaborative learning environment based on intelligent agents. Expert Systems with Applications **14**.
2. Goldman, C.V, and Rosenschein, J.S. (1996) Mutually supervised learning in multiagent systems. In: G. Weiss and S. Sen (eds.), Adaptation and Learning in Multiagent Systems, Springer-Verlag, New York, 85-96.
3. Goldman, C.V., and Rosenschein, J.S. (1997) Multiagent learning systems and expert agents, In: Socially Intelligent Agents, Papers from the 1997 AAAI Fall Symposium (November), 58-60.
4. Huang, M.-J. (1999) Intelligent diagnosing and learning agents for intelligent tutoring systems. Journal of Computer Information Systems (**Fall**), 45-50.
5. Jerbic, B., Grolinger, K., and Vranjes, B. (1999) Autonomous agent based on reinforcement learning and adaptive shadowed network, Artificial Intelligence in Engineering **13**(2).
6. Jouffe, L. (1998) Fuzzy Inference System Learning by Reinforcement Methods, IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews **28** (3).
7. Kokar, M.M, and Reveliotis, S.A. (1993) Reinforcement learning: architectures and algorithms, International Journal of Intelligent Systems **8**(8).
8. Liebowitz, J., Adya, M., Rubenstein-Montano, B., Yoon, Y., Buchwalter, J., Imhoff, M., Baek, S., and Suen, C. (2000). MACS : Multi-Agent COTR System for Defense Contracting, Knowledge-Based Systems Journal **13**(3), 241-250.
9. Maclin, R., and Shavlik, J.W. (1996) Creating advice-taking reinforcement learners, Machine Learning **22**(1-3).
10. Maulsby, D., and Witten, I. H. (1997) Teaching agents to learn: From user study to implementation. IEEE Computer **30**(11), 36-44.
11. Mitchell, T. (1997) Machine Learning, McGraw-Hill: New York.
12. Norrie, D.H., and Gaines, B.R. (1995) The learning Web: A system view and agent-oriented model. International Journal of Educational Telecommunications **1**(1), 23-41.
13. Prasad, M.V.N., and Lesser, V.R. (1999) Learning situation-specific coordination in cooperative multi-agent systems. Autonomous Agents and Multi-Agent Systems **2**, 173-207.
14. Rubenstein-Montano, B., Lowry, S., Cantu, F., Drummey, K., Yoon, V., Wilson, T., and Liebowitz, J. (2001) Agent Learning in the Multi-Agent Contracting officer's technical representative System (MACS). Submitted to Autonomous Agents and Multi-agent Systems.
15. Sen, S., and Sekaran, M. (1996) Multiagent coordination with learning classifier systems. In: Adaption and Learning in Multi-Agent Systems, IJCAI 95 Workshop, Springer-Verlag, Berlin, Germany, 218-233.
16. Shen, W. (1994) Autonomous Learning from the Environment, Computer Science Press, New York.
17. The Open Agent Architecture ™ , (2000), www.ai.sri.com/~oaa.
18. Vaario, J., and Ueda, K. (1996) Modular learning in a multiagent environment. In: C. H. Dagli, M. Akay, C. L. P. Chen, B. R. Fernandez and J. Gosh (eds.) Intelligent Engineering Systems Through Artificial Neural Networks, ANNIE 96 Proc., ASME Press, New York.
19. Yoon, V., Wilson, T., Lowry, S., Rubenstein-Montano, B., and Liebowitz, J. (2001) Natural Language Interface for the Multi-Agent COTR System (MACS). Submitted to Applied AI.
20. Zeng, D., and Sycara, K. (1998) Bayesian learning in negotiation. International Journal of Human Computer Studies **48**(1), 125-141.